CSS Reference

Style responsive and accessible interfaces using reusable CSS rules, layout systems and clear visual states.

Connecting a CSS Stylesheet

CSS controls presentation while HTML provides structure.

Connect the file

Create `css/style.css`, then add this inside the page's `<head>`:

```
<link rel="stylesheet" href="css/style.css">
```

A relative path is interpreted from the current page. A PHP page in a subfolder may need a different path.

Test the connection

```
body {
  font-family: Arial, sans-serif;
  color: #1f2933;
  background: #f7f9fc;
}
```

Refresh without cache if an old version remains.

Good practice

Keep reusable rules in external stylesheets. Avoid large blocks of inline CSS and do not use colour as the only way to communicate meaning.

Check

- [ ] Link is inside `head`.

- [ ] Path matches folder structure.
- [ ] Browser developer tools show no 404.
- [ ] Page remains understandable if CSS does not load.

CSS Selectors, Classes and Reusable Rules

Selectors identify the HTML elements a rule styles.

```
body { margin: 0; }
h1 { color: #123f73; }
.card { border: 1px solid #cbd5e1; }
#main-search { max-width: 40rem; }
nav a:hover,
nav a:focus-visible { text-decoration: underline; }
```

Use element selectors for broad defaults, classes for reusable components and IDs mainly for unique relationships or scripting. Avoid selectors tied to one accidental nesting structure.

```
<article class="card">
  <h2>Team summary</h2>
</article>
```

Cascade and specificity

When rules conflict, source order and specificity decide which applies. Prefer simple classes over repeated `!important`.

Check

- [ ] Class names describe purpose.
- [ ] Repeated designs reuse the same class.
- [ ] Focus styles are not removed.
- [ ] Selectors remain understandable.

CSS Box Model and Spacing

Every element is a box made from content, padding, border and margin.

```
*,
*::before,
*::after {
  box-sizing: border-box;
}

.card {
  max-width: 42rem;
  padding: 1rem;
  border: 1px solid #cbd5e1;
  margin-block: 1rem;
}
```

With `border-box`, declared width includes padding and border.

Use padding inside a component and margin between components. Prefer relative units such as `rem` for scalable spacing.

Avoid fixed layouts

Large fixed widths can overflow small screens. Combine `width: 100%` with a suitable `max-width`.

Check

- [ ] Global border-box rule is present.
- [ ] Spacing has a consistent scale.
- [ ] Content does not touch borders.
- [ ] Page has no horizontal scrolling at narrow widths.

Responsive Layouts with Flexbox and Grid

Responsive layouts adapt to available space rather than one device size.

Flexbox navigation

```
.navigation {
  display: flex;
  flex-wrap: wrap;
  gap: 0.75rem;
  align-items: center;
}
```

Grid cards

```
.card-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(min(16rem, 100%), 1fr));
  gap: 1rem;
}
```

Media query

```
@media (max-width: 40rem) {
  .page-header {
    align-items: stretch;
  }
}
```

Allow text to wrap and avoid fixed heights. Test zoom, long labels and real data.

Check

- [ ] Layout works at narrow and wide widths.
- [ ] Reading order remains logical.
- [ ] Keyboard order matches visual order.
- [ ] Tables have a deliberate small-screen treatment.
- [ ] Controls remain large enough to use.

Accessible Colour, Typography and Focus

Accessible styling supports perception, reading and keyboard use.

```
body {
  color: #1f2933;
  background: #ffffff;
  font-family: system-ui, sans-serif;
  line-height: 1.6;
}

a { color: #075985; }

:focus-visible {
  outline: 3px solid #f59e0b;
  outline-offset: 3px;
}
```

Maintain sufficient contrast and never communicate status through colour alone. Pair colour with text, icons or patterns.

Do not disable zoom, remove focus outlines or use tiny fixed text. Keep line lengths comfortable and headings visually distinct.

Reduced motion

```
@media (prefers-reduced-motion: reduce) {
  *, *::before, *::after {
    scroll-behavior: auto;
    animation-duration: 0.01ms !important;
  }
}
```

Check

- [ ] Text/background contrast is sufficient.
- [ ] Focus indicator is obvious.
- [ ] Status includes words, not colour alone.
- [ ] Page works at 200% zoom.
- [ ] Motion is not required to understand content.

Styling Forms and Validation Messages

Form styling should make labels, controls, instructions and errors easy to identify.

```
.form-group {
  display: grid;
  gap: 0.35rem;
  margin-block: 1rem;
}

input,
select,
button {
  font: inherit;
  min-height: 2.75rem;
}

input {
  border: 1px solid #64748b;
  border-radius: 0.25rem;
  padding: 0.6rem;
}

input:focus-visible {
  outline: 3px solid #f59e0b;
  outline-offset: 2px;
}

.error {
  color: #8a1c1c;
  border-left: 4px solid #b42318;
  padding: 0.75rem;
}
```

Do not rely on placeholder text as the label. Keep validation messages near the relevant control and provide a summary for multiple errors.

Check

- [ ] Every control retains a visible label.
- [ ] Focus and error states are distinct.
- [ ] Buttons look actionable.
- [ ] Messages remain understandable without colour.
- [ ] Keyboard navigation follows a logical order.

Styling Tables, Navigation and Data Displays

Data interfaces should prioritise interpretation over decoration.

```
.table-wrap { overflow-x: auto; }

table {
  width: 100%;
  border-collapse: collapse;
}

th, td {
  padding: 0.65rem;
  border: 1px solid #cbd5e1;
  text-align: left;
}

th { background: #123f73; color: #fff; }

tbody tr:nth-child(even) { background: #f8fafc; }

nav ul {
  display: flex;
  flex-wrap: wrap;
  gap: 0.75rem;
  list-style: none;
  padding: 0;
}
```

Use real table markup for tabular data. A responsive wrapper permits horizontal scrolling without changing table meaning.

Highlight the current navigation link with text/shape as well as colour, and retain a visible focus state.

Check

- [ ] Table has a caption and header cells in HTML.
- [ ] Narrow screens can reach every column.
- [ ] Navigation wraps safely.
- [ ] Current location and focus are identifiable.
- [ ] Empty datasets have a visible message.